**PWM**

O Modulador PWM é um dispositivo ou circuito digital essencial que gera um sinal de onda quadrada, alternando rapidamente entre os estados ligado (nível alto) e desligado (nível baixo) em uma frequência constante. Sua finalidade é simular uma saída de tensão analógica ou controlar a potência média fornecida a uma carga, como um motor ou um LED. A técnica de Modulação por Largura de Pulso (PWM) descreve o método de codificação: a mensagem analógica é incorporada na duração (largura) do pulso ligado. O parâmetro de controle chave é o Duty Cycle (Ciclo de Trabalho), que é a proporção percentual do tempo em que o sinal permanece ativo (ON) dentro de um período total.

Por exemplo, um *Duty Cycle* de 50% significa que o sinal está ligado metade do tempo e desligado a outra metade, resultando em 50% da potência máxima entregue. A grande diferença entre o PWM e um sinal analógico contínuo é que, enquanto o sinal analógico assume qualquer valor de tensão dentro de sua faixa, o PWM é inerentemente digital (apenas ON/OFF). O PWM alcança a mesma função de controle de potência com muito mais eficiência, pois, ao atuar como uma chave digital, minimiza as perdas de energia por calor.

Características Sinal PWM (Modulação por Largura de Pulso) e Sinal Analógico Contínuo.

**Natureza:**

Digita; O sinal tem apenas dois níveis (ON ou OFF, 5V ou 0V, 3.3V ou 0V). Contínua. O sinal pode assumir qualquer valor dentro de uma faixa (ex: 0V, 1.2V, 3.7V, 5V, etc.).

**Controle:**

Digital; Varia a largura do pulso (Duty Cycle) para alterar a potência média. Analógico; Varia a tensão/amplitude em si para alterar a potência.

**Eficiência:**

Digital; altíssima eficiência. Como a saída é apenas uma chave ligada/desligada, há pouca perda de energia na forma de calor. Analógico; Baixa eficiência. Mudar a tensão (como em amplificadores lineares) gera calor significativo.

Aqui está o resumo solicitado, sem o uso de tabelas:

**Aplicação Prática no Arduino (PWM)**

O Arduino gera um sinal de Modulação por Largura de Pulso (PWM) utilizando *timers* (temporizadores) internos. Essa técnica simula uma saída analógica a partir de um pino digital, mantendo a frequência fixa e variando o ciclo detrabalho (duty cycle**)**, que é a proporção de tempo em que o sinal fica em nível lógico alto.

*Exemplo Prático (Controle de Brilho de LED):* Conecta-se um LED a um pino PWM (como o pino 9) e usa-se analog Write() dentro de um laço (for loop) para incrementar e decrementar gradualmente o valor (de 0 a 255 e vice-versa), controlando assim a intensidade luminosa do LED.

**Interferência do PWM no ADC**

O **ADC (Conversor Analógico-Digital)** converte sinais analógicos contínuos de sensores em valores digitais (0 a 1023 no Uno) que o microcontrolador pode processar. A interferência ocorre porque o sinal PWM é uma onda quadrada que chaveia rapidamente. Esta comutação, especialmente ao acionar cargas (como motores), injeta ruído de alta frequência nas linhas de alimentação e terra do microcontrolador. Como o ADC usa a tensão de alimentação ou uma referência para suas medições, esse ruído a desestabiliza, causando leituras analógicas imprecisas, erráticas ou oscilantes.

**As soluções para evitar essa interferência incluem:**

Aplicar filtros passa-baixa RC (Resistor-Capacitor) na entrada analógica para suavizar o sinal e remover ruído acoplado. Utilizar capacitores dedecuplagem/bypass próximos aos pinos de alimentação do microcontrolador para absorver picos de ruído. Implementar a separação de alimentação, utilizando fontes ou trilhas separadas para a parte de potência (PWM) e a parte analógica (ADC/sensores). Entender essa interferência é vital em projetos de IoT para garantir a precisão dos dados de sensores (coletados pelo ADC), assegurando a confiabilidade e a estabilidade do sistema, que frequentemente depende da correta coleta de informações para tomar decisões.